A Comparison of Explicit and Implicit Graph Embedding Methods for Pattern Recognition

Donatello Conte, Jean-Yves Ramel, Nicolas Sidère, Muhammad Muzzamil Luqman, Benoit Gaüzère, Jaume Gibert, Luc Brun, Mario Vento

A Comparison of Explicit and Implicit Graph Embedding Methods for Pattern Recognition

Donatello Conte[1], Jean-Yves Ramel[2], Nicolas Sidère[2], Muhammad Muzzamil Luqman[3], Benoit Gaüzère[4], Jaume Gibert[4], Luc Brun[4] and Mario Vento[1]

[1] Università di Salerno
Via Ponte Don Melillo, 1, 84084 Fisciano(SA) ITALY
{dconte,mvento}@unisa.it
[2] Université François Rabelais de Tours, Laboratoire Informatique (EA6300)
64 Avenue Jean Portalis, 37200 Tours FRANCE
{ramel,sidere}@univ-tours.fr
[3] L3i Laboratory, University of La Rochelle
17042 La Rochelle FRANCE
muhammad_muzzamil.luqman@univ-lr.fr
[4] GREYC UMR CNRS 6072
14000 Caen FRANCE
{benoit.gauzere,jaume.gibert,luc.brun}@ensicaen.fr

Abstract. In recent years graph embedding has emerged as a promising solution for enabling the expressive, convenient, powerful but computational expensive graph based representations to benefit from mature, less expensive and efficient state of the art machine learning models of statistical pattern recognition. In this paper we present a comparison of two implicit and three explicit state of the art graph embedding methodologies. Our preliminary experimentation on different chemoinformatics datasets illustrates that the two implicit and three explicit graph embedding approaches obtain competitive performance for the problem of graph classification.

1 Introduction

Two important challenges related to graphs concern the structural pattern recognition field: first of all, graph based methods like graph matching are computationally demanding hence restricting the application of such methods. Secondly, despite numerous theoretical results on graphs, the graph space has no strong algebraic properties. It is for example not a group nor a vector space. Such a lack of mathematical properties on the graph's space does not allow to readily combine structural and statistical pattern recognition methods.

Graph embedding methods map either explicitly or implicitly graphs into high dimensional spaces hence allowing to perform the basic mathematical computations required by various statistical pattern recognition techniques. Graph embedding methods appear thus as an interesting solution to address graph clustering and classification problems.

The graph embedding methods are formally categorized as implicit graph embedding or explicit graph embedding. The implicit graph embedding methods are based on graph kernels. A graph kernel is a function that can be thought of as a dot product in some implicitly existing vector space. Instead of mapping graphs from graph space to vector space and then computing their dot product, the value of the kernel function is evaluated in graph space. Such an implicit embedding satisfies all properties of a dot product. Since it does not explicitly map a graph to a point in vector space, a strict limitation of implicit graph embedding is that it does not permit all operations that could be defined on vector spaces. Further reading on state of the art methods of graph kernels and implicit graph embedding could be found in [2].

On the other hand, explicit graph embedding methods explicitly embed an input graph into a feature vector and thus enable the use of all the methodologies and techniques devised for vector spaces. The vectors obtained by an explicit graph embedding method can also be employed in a standard dot product for defining an implicit graph embedding function between two graphs. An important property of explicit graph embedding is that graphs of different size and order need to be embedded into a feature vector of determined size. The selection of the axis of this feature vector requires thus a fine analysis of the analysed dataset in order to selected features representive of the set while remaining sufficiently generic to describe any input graph. We refer the interested reader to [16] for further reading on classical explicit graph embedding techniques.

Similarly to the previous study described in [6], in this paper we propose a comparison between two implicit graph embedding methods based on graph kernels ([1, 5]) and three methods of explicit graph embedding with comparable behavior ([7, 12, 17]). The difference between these techniques will be illustrated on classification problems using chemoinformatic datasets, such as those from IAM [14], the predictive toxicology challenge (PTC) dataset [20] and the MAO dataset from GREYC [5]. In Section 2 and Section 3 we describe the respective graph kernels and explicit graph embedding methods. Section 4 details the experimental evaluation alongwith a discussion on the classification performance of these methods. Finally, the paper concludes in Section 5.

2 Classification by Graph Kernels Methods

2.1 Method 1: Laplacian Graph Kernel

The graph edit distance between two graphs is defined as the minimal number of vertices and edge removal/addition/relabeling required to transform one graph into an other [13]. Unfortunately, even though the edit distance defines a metric under weak conditions, this distance is not definite negative. Consequently, kernels directly based on the edit distance are not definite positive and hence do not correspond to a valid kernel.

Let us consider a set of input graphs $\{G_1, \ldots, G_n\}$ defining our graph test database. Given a kernel k, the gram matrix K associated to this database is

an $n \times n$ matrix defined by $K_{i,j} = k(G_i, G_j)$. As denoted by Steinke [19], the inverse of K (or its pseudo inverse if K is not invertible) may be considered as a regularization operator on the set of vector of dimension n. Conversely, the inverse (or pseudo inverse) of any definite positive regularization operator may be considered as a kernel.

From this point view, designing a "good" graph kernel comes up to define for each dataset a Gram matrix K whose associated norm penalizes the mapping of different values to similar graphs. One scheme to design a kernel consists thus to first build a definite positive regularization operator and then to take its inverse (or its pseudo inverse) to obtain a kernel. Let us consider the Laplacian operator defined as follows: given the set of graphs, $\{G_1, \ldots, G_n\}$, we first consider the $n \times n$ adjacency matrix $W_{i,j} = e^{-\frac{d(G_i,G_j)}{\sigma}}$ where σ is a tuning variable and $d(\cdot,\cdot)$ denotes the edit distance [15]. The normalized Laplacian of $\{G_1, \ldots, G_n\}$ is then defined as $L = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ where D is a diagonal matrix defined by $D_{i,i} = \sum_{j=1}^{n} W_{i,j}$.

Well known results from spectral graph theory ([3]) establish that L is a symmetric, semi definite positive matrix whose eigenvalues belongs to the interval $[0, 2]$. Unfortunately, the Laplacian is also well know for being non invertible since the eigenvector vector $\mathbf{1} = (1, \ldots, 1)$ is associated to the eigenvalue 0. The only semi definite property of the Laplacian matrix forbids a direct inversion of this matrix. Moreover, the pseudo inverse of the Laplacian induces numerical instabilities which does not lead to a reliable kernel. Therefore, following Smola [18], we rather regularize the spectrum of L. The regularized version of L, denoted as $\tilde{L}$, is defined as $\tilde{L} = I + \lambda L$, where λ is a regularisation coefficient. The regularized laplacian $\tilde{L}$ is invertible and its inverse $K = \tilde{L}^{-1}$ is taken as a kernel. Using a classification or regression scheme, such a kernel leads to map to close values graphs having a small edit distance (and thus a strong similarity).

The implicit embedding induced by the Graph Laplacian kernel is not fixed by some a priori rules but is deduced from a regularization of the matrix of pairwise distances between objects. From this point of view, this implicit embedding is close from the explicit embedding proposed by Jouili and Tabbone [10] which additionally requires a dimensionality reduction step.

2.2 Method 2: Treelet Kernel

Treelet kernel [5] is a graph kernel based on a bag of non linear patterns which computes an explicit distribution of each pattern within a graph. This method explicitly enumerates the set of treelets included within a graph. The set of treelets, denoted $\mathcal{T}$, is defined as the 14 trees having a size lower than or equals to 6 nodes. Thanks to the limited number of different patterns encoding treelets, an efficient enumeration of the number of occurrences of each labeled pattern within a graph can be computed by algorithm defined in [5]. Labeling information included within treelets is encoded by a canonical key which is defined such as if treelets have a same structure, their canonical key is similar if and only if the two treelets are isomorphic. Each treelet being uniquely identified by the index

of its pattern and its canonical key, any graph G can be associated to a vector $f(G)$ which encodes the number of occurrences of each treelet $t \in \mathcal{T}$ by $f_t(G)$. Note that this vector representation may be of very high dimension since it may encode all possible treelets according to all possible nodes and edges labellings defined for a graph family. In chemoinformatics, such a vector representation may have a dimension higher than 4.25×10^9 [5] which forbids its explicit vector embedding. Treelet kernel between graphs is defined as a sum of sub kernels between common treelets of both graphs:

$$K_{\mathcal{T}}(G, G') = \sum_{t \in \mathcal{T}(G) \cap \mathcal{T}(G')} k(f_t(G), f_t(G')) \quad (1)$$

where $\mathcal{T}(G)$ encodes the set of treelets included within G and $k(.,.)$ defines any positive definite kernel between real numbers such as linear, Gaussian or polynomial kernel. Each sub kernel $k(.,.)$ encodes the similarity of the number of occurrences for each treelet t common to both graphs to be compared.

In order to improve the accuracy of treelet kernel, each treelet can be weighted according to a prediction task:

$$K_{\mathcal{T}}(G, G') = \sum_{t \in \mathcal{T}(G) \cap \mathcal{T}(G')} w(t) k(f_t(G), f_t(G')) \quad (2)$$

As described in [4], each weight $w(t) \in \mathbb{R}^+$ can be computed in a sparse and optimal way for a given training set by using multiple kernel learning (MKL). Using sparsity promotes the selection of relevant treelets according to the prediction task and $w(t)$ can thus be understood as a measure of the importance of treelet t for the prediction task.

3 Classification by Graph Embedding Methods

3.1 Method 1: Topological Embedding

One of the main challenges of graph embedding is to preserve topological information provided by the graph representation after transformation into a feature vector. The topological embedding method proposed in [17] provides an interesting answer to this problem by using a generic lexicon of topological structures that could be enumerated in graphs during the computation of the vectorial signature of the graphs. However, this lexicon must be comprehensive enough to ensure discrimination from a graph from another. They have therefore decided to take as a baseline the non-isomorphic graphs network presented in [9]. The network presents all graphs composed of n edges up to N (where N is the maximum number of edges). Thereafter, the term pattern will refer to a subgraph element of the non-isomorphic graph network.

For example, Figure 1 shows the non-isomorphic graph network until the fourth rank giving a lexicon of 11 patterns.

The vectorial representation of a graph topology will be built by counting the occurrences of each pattern of the lexicon. In other way, each element of the

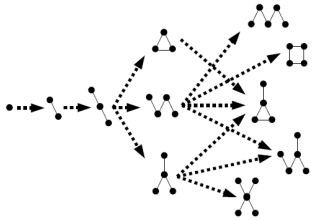
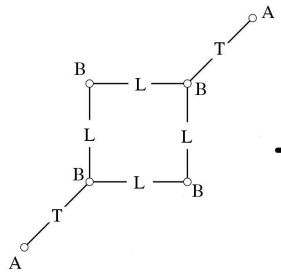


Fig. 1. The non-isomorphic graph network used to embed the topology.



topology	6	6	8	8	0	2
vertices A	2	2	4	4	0	2
vertices B	4	10	20	28	0	6
edges T	0	2	4	4	0	2
edges L	0	4	12	20	0	4

(b)

Fig. 2. Matrix (b) corresponding to vectorial signature of graph presented in (a).

vector is the frequency of apparition of a pattern, which represents a descriptor of a part of the graph. Thus, the topology of the graph is embedded in the vectorial representation. This vectorial representation needs now to be enriched by encapsulating the information provided by labels that can be associated to the edges and vertices. As each of these labels can be composed with several attributes, the inclusion of this information can be problematic regarding the nature (numerical) and the number of attributes constituting a label. Two ways are proposed to by-pass these problems :

1. the first method consists of discretizing numerical attributes to obtain symbolic attributes. Then a combination of all these symbolic attributes can be realized to list all possible labels.
2. the second method is to perform a clustering in the label space using attributes as feature vectors. This results in some new classes of labels where their number can be controlled.

The construction of the vectorial representation can then be performed by filling all the cells of the matrix generated with the frequency of each pattern and each label for this pattern. The matrix (see Fig. 2b) presents an example of the proposed vectorial representation for the graph represented on Fig. 2a. More details about this topological embedding method can be found in [17].

6

3.2 Method 2: Fuzzy Multilevel Graph Embedding (FMGE)

The Fuzzy Multilevel Graph Embedding method (FMGE) performs multilevel analysis of graph to extract discriminatory information of three different levels. These include the graph level information, structural level information and the elementary level information. The three levels of information represent three different views of graph for extracting global details, details on topology of graph and details on elementary building units of graph. The feature vector of FMGE is named Fuzzy Structural Multilevel Feature Vector - FSMFV (see Fig. 3).

Graph order	Graph size	Embedding of node degree	Embedding(s) of subgraph(s) homogenity	Embedding(s) of node attribute(s)	Embedding(s) of edge attribute(s)

Fig. 3. Feature vector of FMGE.

The features for graph level information represent a coarse view of graph and give general information about the graph. These features include graph order and graph size.

The features for structural level information represent a deeper view of graph and are extracted from the node degrees and subgraph homogeneity in graph. Subgraph homogeneity is represented by computing resemblance attributes for the nodes and edges of graph. The resemblance attributes for an edge is computed from the attributes on its neighboring nodes. The resemblance for a numeric attribute (a) is computed as a ratio of this attribute's values on neighboring nodes of an edge (a_1 and a_2) (see Eq. 3). Whereas the resemblance for a symbolic attribute (b) is computed as a ratio of this attribute's values on neighboring nodes of an edge (b_1 and b_2) (see Eq. 4).

$$resemblance(a_1, a_2) = min(|a_1|, |a_2|)/max(|a_1|, |a_2|) \tag{3}$$

$$resemblance(b_1, b_2) = \begin{vmatrix} 1 & b_1 = b_2 \\ 0 & otherwise \end{vmatrix} \tag{4}$$

The third level of information is extracted by penetrating into further depth and more granular view of graph and employing details of the elementary building blocks of graph. These features represent the information extracted from the node and edge attributes. The node degree, numeric resemblance attributes, numeric node attributes and numeric edge attributes are embedded by fuzzy histograms whereas the symbolic resemblance attributes, symbolic node attributes and symbolic edge attributes are embedded by crisp histograms. FMGE learns the intervals, for constructing these histograms, during an unsupervised learning phase and employs the learned intervals during graph embedding phase [12].

The feature vector obtained by FMGE is based on histogram encoding of the multilevel information extracted from graph. The number of features in the

vector is directly dependent on the number of bins employed for constructing these histograms. The use of high dimensional histograms is explicitly built into the method as it enables FMGE to provide a more robust encoding of information and enables it to generalize to unseen graphs. However, the feature vector can become sparse and confuse between classes of graphs. In order to reduce the size of FMGE feature vector and to remove the unimportant features for a given graph dataset, we select the subset of top-ranked features on the basis of ranks obtained through the Relief algorithm [11].

3.3 Method 3: Attribute Statistics based Embedding

The attribute statistics based embedding of graphs is a simple and efficient way of expressing the labelling information stored in nodes and edges of graphs in a rather naive feature vector. It basically consists in computing frequencies of appearences of very simple subgraph structures such as nodes with certain labels or node-edge-node structures with specific label sequences. Formally, consider a set of graphs $\mathcal{G} = \{g_1, \ldots, g_N\}$, with $g_i = (V_i, E_i, \mu_i, \nu_i)$ being the ith graph in the set with labelling alphabet L_{V_i} for the nodes and L_{E_i} for the edges. We assume that all graphs in $\mathcal{G}$ have the same labelling alphabets, this is $L_{V_i} = L_{V_j}$ and $L_{E_i} = L_{E_j}$ for all $i, j \in \{1, \ldots, N\}$. We do not assume, however, that each node and edge label occurs in each graph. Let $L_V = \{\alpha_1, \ldots, \alpha_p\}$ and $L_E = \{\omega_1, \ldots, \omega_q\}$ be the common labelling alphabets.

For each graph $g = (V, E, \mu, \nu) \in \mathcal{G}$, we define p unary features measuring the number of times each label in L_V appears in the graph, this is

$$U_i = \#(\alpha_i, g) = |\{v \in V \,|\, \alpha_i = \mu(v)\}|. \tag{5}$$

Binary features for edges are defined by computing how many times each possible sequence of node-edge-node labels appears in the graph. In particular,

$$\begin{aligned} B_{ij}^k &= \#\left([\alpha_i \leftrightarrow \alpha_j]_{\omega_k}, g\right) \\ &= |\{e = (u, v) \in E \,|\, \alpha_i = \mu(u) \wedge \alpha_j = \mu(v) \wedge \omega_k = \nu(e)\}|. \end{aligned} \tag{6}$$

Note that, since graphs are undirected, these features are symmetric, this is, $B_{ij}^k = B_{ji}^k$ for all $i, j \in \{1, \ldots, p\}$. We can then just consider half of them and always assume that $i \leq j$. This results in defining $\frac{1}{2} \cdot q \cdot p \cdot (p+1)$ binary features.

The final embedding configuration is the ensemble of all this features. Note than another interpretation of these features is their relation with random walks. In particular the random walk graph kernel implicilty computes the number of random walks of any length in each graph. In the attribute statistics based embedding case, one just considers walks of length 0 (node labels appearences) and walks of length 1 (node-edge-node label sequences). Although much simpler and local, the fact that these features are explicitly built makes them interesting and flexible enough to provide robust results in several classification problems. Distance correlation with edit distance or their extension to continuous attributed graphs [8, 7] have also been shown in the literature.

4 Experimental Results

This section deals with the experimentation aiming at evaluate and compare the implicit and explicit embedding approaches. In particular, we consider classification tasks applied to chemoinformatics datasets.

4.1 The Considered Application and the Dataset

We have conducted experiments on four datasets of molecules. Molecules are easily converted into graphs by representing atoms as nodes and the covalent bonds as edges. Nodes are labeled with chemical symbols and edges by the valence of the linkage:

AIDS This dataset consists of two classes (*active, inactive*) of 2000 graphs representing molecules with activity against HIV or not.

Mutagenicity This dataset is divided in two classes regarding the mutagenicity (one of the numerous adverse properties of a compound that hampers its potential to become a marketable drug) of 4337 molecules.

Predictive Toxicology Challenge (PTC) This dataset deals with the predicting of the outcome of biological tests for the carcinogenicity of chemicals using information related to chemical structure only (*positive or negative*) on four catgories of animals : female rats (FR), male rats (MR), female mice (FM), male mice (MM) with about 240 graphs per set.

Monoamine oxidase dataset (MAO) This problem is defined on a set of 68 molecules divided into two classes: the molecules that inhibit the monoamine oxidase (antidepressant drugs) and those that do not.

These datasets are issued from public repositories. *AIDS* and *Mutagenicity* come from the IAM database repository[5], while *PTC* and *MAO* are both available in the GREYC's Chemistry databank[6]. Classification accuracy is measured by following the classification scheme designed by the datasets authors ([14, 5]). For *AIDS* or *Mutagenicity*, a validation subset is used to optimize an SVM and the classification accuracy is obtained on an independant test subset. We used a k-fold cross-validation approach for PTC (k=68) and MAO (k=10) to parameterize an SVM and obtain the classification mean rates.

4.2 Results and Comparison

Table 1 shows the classification rates achieved by the 6 methods. Taking into account the four datasets, all implicit and explicit methods seems to be competitive and comparable. Of course, depending on the data, some variations can appear but these variations are small and they rather not be a criterion for choosing one method over another. Thus, other considerations should be taken into account.

[5] http://www.iam.unibe.ch/fki/databases/iam-graph-database
[6] https://brunl01.users.greyc.fr/CHEMISTRY/index.html

Table 1. Classification results for different methods and datasets.

				PTC			
	Mutagenicity	AIDS	MAO	FM	MM	FR	MR
Laplacian kernel	70.2	92.6	90.0	59.2	55.2	57.7	60.9
Treelet kernel	77.1	99.1	91.2	58.7	61.9	60.4	60.8
Treelet kernel with MKL	77.6	99.7	94.1	64.2	64.6	71.2	64.8
Topological Embedding	77.2	99.4	91.2	65.9	67.5	68.7	63.7
FMGE	76.5	99.0	92.1	63.9	66.3	60.0	59.9
Attribute Statistics	76.5	99.6	90.6	64.8	63.1	67.9	59.7

In particular, computational complexity or parameterization dependancy should be evaluated for all these approaches.

On top of these things, an important remark one should be aware of is the fact that most of the discussed methodologies might present some restrictions in order to be evaluated in other pattern recognition problems. For instance, explicit embeddings may be favored over implicit ones whenever the explicit vector representation is required by some algorithms which require more than the dot product. Indeed, graph kernels are limited to kernel methods such as SVM. On the other hand, though, implicit ones are usually defined between two graphs whereas most of explicit methods require the whole dataset to compute the graph embedding. So, implicit methods may be favored other explicit one whenever the access to the whole dataset is limited.

5 Conclusions

Graph embedding for pattern recognition is a recent emerging trend to enable the pattern recognition community to benefit from the representative power of graph based structural approaches of pattern recognition and the computational power of machine learning models of statistical pattern recognition approaches. We have outlined two graph kernel based implicit graph embedding methods and three explicit graph embedding methods. Our initial experimentation on different chemoinformatic databases for the problem of graph classification illustrates that all the methods under consideration obtain competitive performance in terms of classification rates. Our future research goals are to take forward this study on the comparison of implicit and explicit graph embedding methods for revealing the strengths of these methods in terms of learning abilities, automatic parameter optimization, computational complexity and other interesting criteria.

References

1. Brun, L., Conte, D., Foggia, P., Vento, M.: A graph-kernel method for re-identification. In: Kamel, M., Campilho, A.C. (eds.) ICIAR (1). LNCS, vol. 6753, pp. 173–182. Springer (2011)

2. Bunke, H., Riesen, K.: Recent advances in graph-based pattern recognition with applications in document analysis. Pattern Recognition 44(5), 1057–1067 (2011)
3. Chung, F.R.K.: Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92). American Mathematical Society (February 1997)
4. Gaüzére, B., Brun, L., Villemin, D.: Graph kernels based on relevant patterns and cycle information for chemoinformatics. In: Proceedings of ICPR 2012. pp. 1775–1778. IAPR, IEEE (November 2012)
5. Gaüzére, B., Brun, L., Villemin, D.: Two new graphs kernels in chemoinformatics. Pattern Recognition Letters 33(15), 2038 – 2047 (2012)
6. Gazre, B., Hasegawa, M., Brun, L., Tabbone, S.: Implicit and explicit graph embedding: Comparison of both approaches on chemoinformatics applications. In: Gimelfarb, G., Hancock, E., Imiya, A., Kuijper, A., Kudo, M., Omachi, S., Windeatt, T., Yamada, K. (eds.) Structural, Syntactic, and Statistical Pattern Recognition, LNCS, vol. 7626, pp. 510–518. Springer Berlin Heidelberg (2012)
7. Gibert, J., Valveny, E., Bunke, H.: Graph embedding in vector spaces by node attribute statistics. Pattern Recognition 45(9), 3072–3083 (2012)
8. Gibert, J., Valveny, E., Bunke, H., Fornés, A.: On the correlation of graph edit distance and l 1 distance in the attribute statistics embedding space. In: Gimel'farb, G.L., Hancock, E.R., Imiya, A., Kuijper, A., Kudo, M., Omachi, S., Windeatt, T., Yamada, K. (eds.) SSPR/SPR. LNCS, vol. 7626, pp. 135–143. Springer (2012)
9. Jaromczyk, J., Toussaint, G.: Relative neighborhood graphs and their relatives. In Proceedings of the IEEE (1992)
10. Jouili, S., Tabbone, S.: Graph embedding using constant shift embedding. In: Proceedings of the 20th International conference on Recognizing patterns in signals, speech, images, and videos. pp. 83–92. ICPR'10, Springer-Verlag, Berlin, Heidelberg (2010), http://dl.acm.org/citation.cfm?id=1939170.1939183
11. Luqman, M.M., Ramel, J.Y., Llados, J.: Improving Fuzzy Multilevel Graph Embedding through Feature Selection Technique. In: Structural, Syntactic, and Statistical Pattern Recognition. vol. 7626, pp. 243–253 (2012)
12. Luqman, M.M., Ramel, J.Y., Lladós, J., Brouard, T.: Fuzzy multilevel graph embedding. Pattern Recognition 46(2), 551–565 (2013)
13. Neuhaus, M., Bunke, H.: Bridging the Gap Between Graph Edit Distance and Kernel Machines. World Scientific Publishing Co., Inc., River Edge, NJ, USA (2007)
14. Riesen, K., Bunke, H.: Iam graph database repository for graph based pattern recognition and machine learning. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T.Y., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) SSPR/SPR. LNCS, vol. 5342, pp. 287–297. Springer (2008)
15. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image Vision Computing 27(7), 950–959 (2009)
16. Riesen, K., Bunke, H.: Graph classification based on vector space embedding. IJPRAI 23(6), 1053–1081 (2009)
17. Sidere, N., Héroux, P., Ramel, J.Y.: Vector representation of graphs: Application to the classification of symbols and letters. In: ICDAR. pp. 681–685. IEEE Computer Society (2009)
18. Smola, A.J., Kondor, R.I.: Kernels and regularization on graphs. In: XV Annual Conference on Learning Theory. pp. 144–158 (2003)
19. Steinke, F., Schlkopf, B.: Kernels, regularization and differential equations. Pattern Recognition 41(11), 3271 – 3286 (2008)
20. Toivonen, H., Srinivasan, A., King, R., Kramer, S., Helma, C.: Statistical evaluation of the predictive toxicology challenge 2000–2001. Bioinformatics 19(10), 1183–1193 (2003)